Universidad Zaragoza

Trabajo Fin de Grado

Multi-Attention Network for One-Shot Object Recognition

Autor/es

Carlos Tierno Jiménez

Director/es

ASM Shihav

Javier Fabra Cano

Escuela de Ingeniería y Arquitectura

2019

B.Sc.Eng. Thesis
Bachelor of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Multiattention network for one shot object recognition

Carlos Tierno Jiménez (s182119)

Kongens Lyngby 2019

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

This project aims to create a model that can perform one-shot object recognition. I.e. Given a dataset with n classes C, the model will be able to recognise a class C_{n+1} never seen before by the model, by giving him only one fair representative example of that new class.
The model presented here is able to categorise an object by distinguishing it with just one example; where a Siamese model and a KNN classification based model have been implemented and compared.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a B.Sc. in Computer Science degree in Zaragoza University.
This thesis was introduced by Uizard Technologies for dealing with the issue of efficiently training models for object recognition purposes.

Carlos Tierno Jimenez (s182119)

Acknowledgements

The thesis was made in collaboration with Uizard Technologies and ASM Shihav from the Computer Vision Department of the Technical University of Denmark (DTU).

I would like to thank Javier Fuentes Alonso and Arturo Arranz Mateo from Uizard, for introducing me to this topic and the guidance during this process, and ASM Shihav, for advice and support whenever I have needed it. But also I would like to express my infinite gratitude to Ignacio Agudo, for the constant support and for being always by my side, and to Andrea Calatayud.

Contents

CHAPTER 1

Introduction

Object recognition is a subtopic of computer vision which involves Image Classification and Object Location. This field aims to give a solution to detect one or more objects in an image, providing not only the class but the its location in the frame.

Due to the increasing interest of this topic in research, the project aims to understand one-shot object recognition techniques and compare the different existing classification models. The milestones in this project are: to study object recognition techniques based on deep learning, to review literature on One-shot learning, to select a dataset, to evaluate the results obtained using two different classification techniques with one-shot and also highlight the difference between one-shot and few-shot results.

By following these steps, this project aims to create a model that can differentiate n classes from a dataset C from a class C_{n+1} that has only been provided with one example for training. In order to achieve this goal, two models has been implemented and analyzed: One of them is a Siamese network classifier, whilst the other is a ResNet18 model, pre-trained with ImageNet, from which the feature vector has been extracted in order to feed a KNN classifier.

The idea of checking if these two different classification models where extracting different features came up, therefore the feature vector of both models has been extracted. This was done by feeding a KNN classifier to assess the complementarity between both models, but processing them previously by PCA, to eliminate redundancies inside the extracted features.

CHAPTER 2

Deep Learning

Deep Learning is a sub-field of machine learning aiming to recreate the brain's functions and structure using algorithms inspired by them, called artificial neural networks. Deep learning algorithms are similar to the structure of the nervous system, where all neurons are connected and exchange information between them.

The difference between machine learning and deep learning can be seen when we observe that in a deep learning model the feature extraction is made by the model itself, whilst the machine learning model relies on previously extracted features.

As an introduction to deep learning; neural networks, convolutional neural networks, recurrent networks, residual networks and object detection methods will be explained.

2.1 Basics

Perceptron:

Figure 2.1: Biological neuron and artificial perceptron.

A perceptron is an abstract version of a biological neuron. It is made up of two types of variables: a bias value and weights. The latter control the importance given to the information collected through the inputs. The output follows the following formula:

$$Output = W \times in_i + bias \tag{2.1}$$

.

Then, the output of the perceptron is transformed by an activation function giving the non-linearity property to it.

The activation function:
The activation function draws the linear output from a perceptron and performs a non-linear transformation on it. This enables the deep learning model to extract non-linear features, which allows the representation of any arbitrary complex function which maps inputs to outputs.
We can find a big amount of functions which serve for this purpose, but only the most popular and also the ones used in this thesis are explained below:

– **Sigmoid**

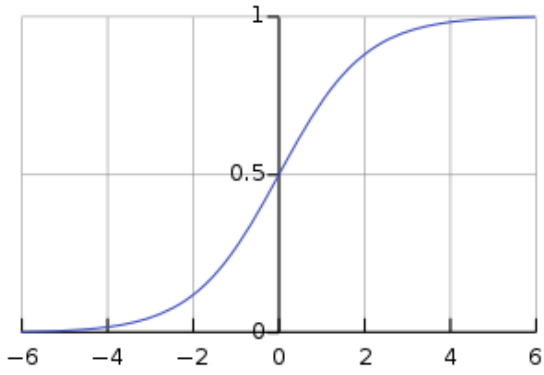

Figure 2.2: Sigmoid representation function: [10] .

$$F(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.2}$$

This function gives an output ranging between 0 and 1. Even thought, it is easy to understand and apply, its popularity is not so high due to the *Vanishing gradient* and optimisation problems. These problems are inflicted by having an output that is not zero centred, which also provokes the saturation of the gradients and a slow convergence.

– **Hyperbolic Tangent (TanH)**
It solves most of the drawbacks of the sigmoid function, by returning an output between -1 and 1. Therefore, optimisation becomes easier although it still suffers from the *Vanishing gradient* problem.

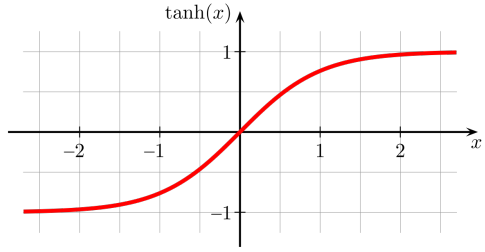

Figure 2.3: Tanh function representation:[9].

$$F(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.3}$$

Previously, the Vanishing gradient problem has been mentioned. The vanishing gradient problem appears when the output space is small compared to the input. Therefore, a significant change in the input would only cause a small change in the result; this happens in the previous functions as the outputs have a range between 0 and 1 or -1 and 1.

– **Rectified Linear Unit (ReLU)**
By using this function, the Vanishing problem is solved. It has been proved that it has six times better convergence than the Tanh function. ReLU follows the following equation:

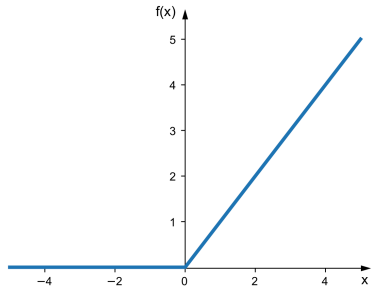

Figure 2.4: ReLU function representation function: [6] .

$$R(x) = max(0, x) \tag{2.4}$$

Neural network: A neural network is made up of a big amount of interconnected perceptrons [2.1]. Each perceptron is set with weights and bias values updated during the training routine based on the error or, more commonly named, loss method, which calculates the difference between the expected and the predicted output.
A simple representation of a perceptron from a hidden layer is shown in Figure 2.1. Every neural network needs to be trained before being used to predict results. In this process, we can identify the following terms:

– **Epoch:** The term that names a whole iteration in which the model goes through the two stages; training and evaluation phases.
– **Batch:** The number of examples processed before fitting the model to adjust weights and biases to achieve higher accuracy.
– **Loss:** Value obtained by calculating the difference between the output expected and the output returned by the model. This is calculated with a Loss function and depending on the type of problem we are solving, we can use different ones.
A simple example of the loss function can be the Mean Squared Error (MSE), which is mathematically defined by:

$$MSE = \frac{1}{n \cdot sum_{n,i=1}((True_i - Predicted_i)^2)} \tag{2.5}$$

During the training of a Neural network, the loss function is used to penalize predicted values which differ from the true value. Achieving the best accuracy is a goal that almost every function tries to reach. However, the purpose can differ between them.

Looking at the MSE function [2.5], can be noticed that the purpose of this function more on Regression problems. The MSE function penalizes the distance between the returned and expected output. This seems reasonable when the problem requires to get values on a scale and giving a value closer or farther have any influence on how good the precision of the model is . For this reason, the classification problems have different loss functions to the regression ones, as misclassifying a 1 as a 2 should not be punished in the same degree as categorizing a 1 as a 5.

Decreasing the model loss is the main goal of the training, and **Optimisation algorithms** helps to minimise (or maximise) this objective function. This optimisation algorithm is a mathematical function dependent on the model's internal learnable parameters.
Two types of optimisation algorithms can be found:

First order algorithms: [7]
This type of algorithms aim to use minimisation on the loss function using its gradient to modify the internal parameters. An example of FoA is Gradient Descent. This algorithm uses the first derivative function, deciding whether the function decreases using the information calculated from partial derivations of the loss functions.

Second order Algorithms:
The second-order method takes advantage of the Hessian to minimise the Loss function. The Hessian is a matrix of second partial derivatives, which works with them to deduce the direction where the first derivative decreases, and, therefore, be able to point out the minimum in an easier way. Although the second-order method gives a much better solution, the time and memory consumed when the derivative is not known, is high. Therefore, first order algorithms are more commonly used.

The gradient descent algorithm is an essential optimisation technique described mathematically by the expression:

$$\theta = \theta - \eta \cdot \bigtriangledown J(\theta) \tag{2.6}$$

where η is the learning rate, and $\bigtriangledown J(\theta)$ is the gradient of the Loss function.

This algorithm performs weight updates in the Neural network model through backpropagation. The backpropagation task is done after calculating the dot

product of inputs signals and their corresponding weights and applying the activation function; carrying the error terms through the network and updating weights.

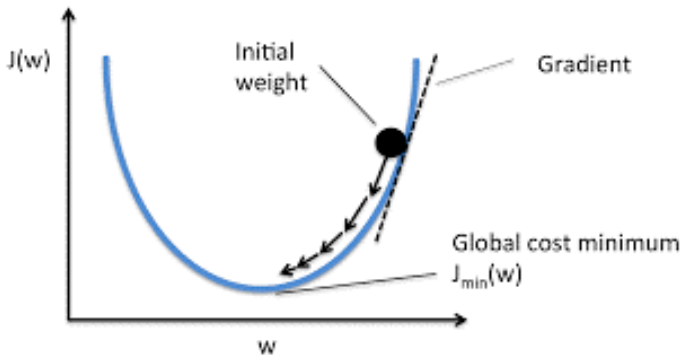

Figure 2.5: Gradient descent graphic[8].

Stochastic gradient descent uses the gradient descent algorithm adding x(i) and y(i) to the cost function terms used in the gradient, resulting in the following function:

$$\theta = \theta - \eta \cdot \bigtriangledown J(x(i), y(i), \theta) \tag{2.7}$$

Due to the new terms added, frequent updates occur forcing the loss function to fluctuate in different intensities, improving the discovery of new and better local minima.
In this thesis, the Adam method is used. A method which uses Adaptive moment estimation. This method applies adaptive learning rates for each parameter, in addition to an exponentially decaying average of past gradients, similar to momentum. Adam works well in practice as it rectifies every problem faced in optimisation techniques such as vanishing Learning rate, slow convergence or high variance in the parameter updates.

The following image shows the convergence through the time of the different algorithms, where can be seen that Adam finds the minimum faster than others (yellow line).

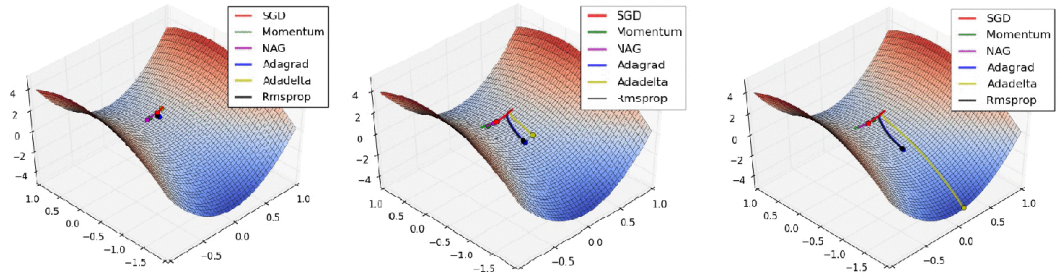

Figure 2.6: Adam and SGD progression [7].

2.2 Convolutional Neural Networks - CNN

Convolutional neural networks (CNN) are mainly used for processing image inputs. This network aims to reduce the images into a form which is easier to process without losing the main features, which are critical to obtain the right prediction.

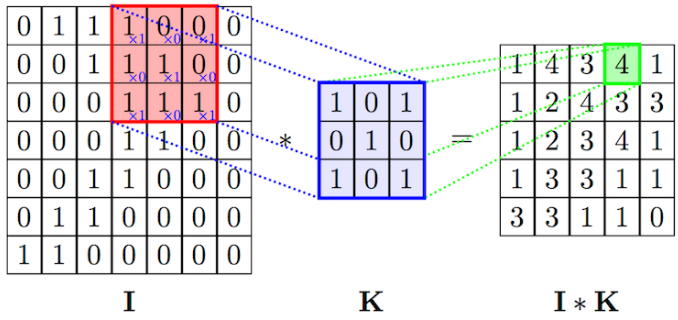

Figure 2.7: Visual explanation of how Convolutional layer works [1] .

In a convolutional layer, three elements configure itself:

- **The kernel**, involved in transforming the image to extract the features. The Kernel acts like a filter which goes through the whole image giving the result value as the product of multiplying a square from the source with the kernel.

- **The Stride**, establishes the times the filter moves to the right (or down if it is at the right part of the matrix)
- **The padding** is used in the convolution layer to control the size of the output. Using the Figure 2.7 as reference, we can see that the source matrix dimensions are reduced when going through the convolution layer. This relation is given by the following formula:

$$OutputDim = \frac{N - F + 2P}{S} + 1 \tag{2.8}$$

  Where N is the size of input (NxN), F is the size of Kernel (FxF), S the stride and P is the padding. Usually, Zero-padding is used to maintain the size of the input but it can also be used as a pooling layer in which the size is reduced in a 2:1 scale.
  Usually, Zero-padding is used to maintain the size of the input but also can be used for acting as a pooling layer in which the size is reduced in 2:1 scale.

A layer type commonly used in CNN models is the pooling layer. This layer aims to reduce the dimensions of the data by combining the outputs of one layer into one unique input. Hence, the output is a scaled representation of the previous.

2.3 Residual Network

CNN is an architecture that has evolved over the years. The first architecture began with AlexNet, which had only five convolutional layers. This has evolved to 19 Convolutional Layers, named VGG network, and 20, named GoogleNet. Despite the increase in the number of layers, the result did not vary much. So, to solve the problem, Residual Networks appeared.
ResNet is a Residual Network model that bases its main idea on mixing the output with a previous input for providing the output layer with a referenced function, instead of learning unreferenced ones.

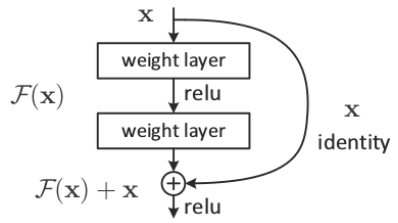

Figure 2.8: Residual basic unit [2].

With this modification the model achieves an improvement that would give it significant popularity among the research community, causing it to appear in different versions of the model: Resnet50, Resnet101, ResnetXt, etc.

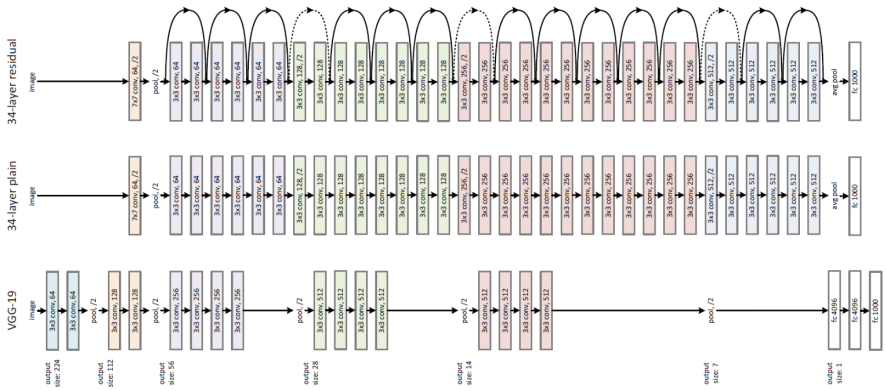

Figure 2.9: ResNet 18 vs Plain model.

CHAPTER 3

Related Work

3.1 Siamese Networks for One-shot Image Recognition

[4] Siamese networks were introduced in the early 1990s by Bromley and LeCun to solve signature verification. A Siamese neural network consists of twin networks which have different inputs but are joined together by a distance function at the end. Authors of Siamese networks, Bromley and LeCun, used a contrastive function to decrease the dissimilarity score of matching pairs, and increase it for unlike pairs.

Nonetheless, in this paper [**siameseppr**], L1 is used, which minimizes the absolute differences between the estimated values and the existing target values. These computed distance between feature vectors is used combined with sigmoid activation.

The process of learning useful features can be computationally expensive and may encounter difficulties in cases where little data is available. With this network Gregory Koch, Richard Zemel and Ruslan Salakhutdinow aim to capitalise on powerful discriminative features to generalise the predictive power of the network to new data: new classes from new distributions using convolutional architectures.

The model instantiated by this paper works over an inner model focused on a verification task. For this purpose, the following model is implemented:

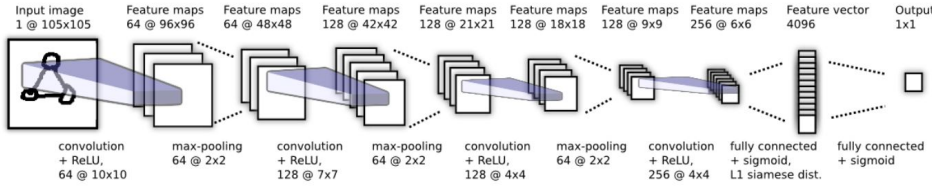

Figure 3.1: Innner model of the Siamese Network.

The model displayed in the previous image [3.1], is the representation of the inner network. The siamese network representation is not shown but the twin network will be combined right after the 4096 unit fully-connected layer, where the L1 distance is computed.
In this paper, momentum optimisation, weights and biases initialisation, learning scheduler, and Bayesian hyperparameter optimisation, are used for improving the model convergence.

Weight and bias initialization is made with different distributions for each of them, discriminating also, by type of layers; The initialization values are set as following:
It is used a zero-mean and a standard deviation of 10^{-2} for weights and a 0.5 mean and a standard deviation of 10^{-2} for biases initialization in convolutional layers. For fully connected layers the biases are initialized in the same way as in convolutional layers, but weights have a zero-mean, and 2×10^{-1} the standard deviation applied.

3.2 One Shot Learning

One shot Learning is the attempt to build models that are able to learn a class by just looking at one sample. This tries to break totally with the current idea of the Artificial Inteligence fields, where thousands of samples are needed to aim a useful model.

The model Google DeepMind [5] implements finds the inspiration on models such as sequence to sequence(seq2seq) with attention, memory networks and pointer networks. The implementation can be divided in two parts: A attention mechanism which computes a small dimensional embedding of an input example and n attention mechanism which compares the input sample with the samples already known.
Also data augmentation has been probed of being a good mechanism for improving one-shot learning models. This also increase the dataset memory load but involves easy transformations like distortion, translation or rotation of the dataset samples.

CHAPTER 4

One-Shot Object recognition

Object recognition is known for needing thousands of examples of objects to learn how to identify a class in a never seen environment. Transfer learning has improved this problem, but the idea of a model recognizing new classes with just a single example has been a trending topic in the Deep Learning community.

The models presented down below aim to compare two different methods of One-shot classification. Therefore, three models have been created to evaluate the differences between Siamese networks and KNN classification:

4.1 Siamese network

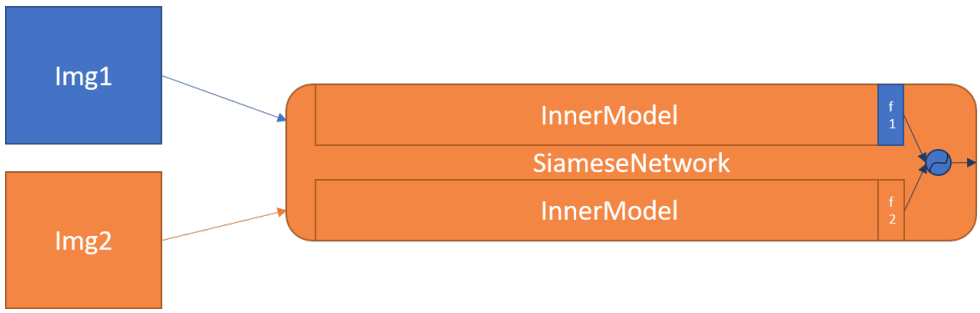

Figure 4.1: Siamese Network.

The Figure 4.2 introduces the structure of the Siamese model. For this model, two different examples have to be given as input. These two images will be processed by the inner model, which will extract the features of the images, to be compared

and to extract a similarity score.

The extraction and comparison are done by the following inner model, which is a replica of the one used in the Paper [**url:GoogleDM**]

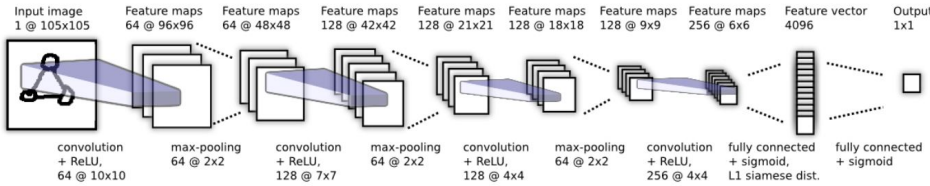

Figure 4.2: Inner Model of the Siamese Network.

4.2 ResNet18 model with KNN Classifier

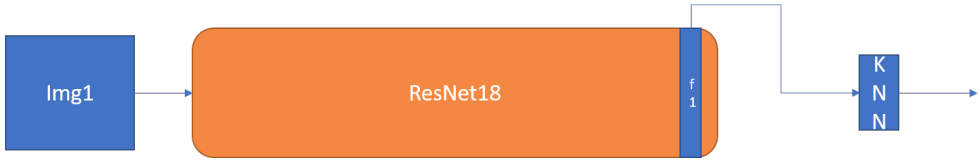

Figure 4.3: ResNet 18 with KNN Classifier.

Figure 4.3 presents the ResNet model, which bases its one-shot classification task on a KnnClassifier. The ResNet is a ResNet18 network pre-trained on ImageNet that has been modified to extract the resulting features of the last layer before the classification task is completed.

The KNN classifier is a 1 near neighbour classifier as it can just be trained with 1 example per class.

4.3 Cross-entropy loss

Cross-entropy Error is usually used for binary classification where the neural model has one single logistic unit as an output layer. With this combination, the output

will become a value between zero and one interpreted as a probability.

Both models described above use this loss function in different ways:

- The Siamese model compares two feature vectors from different inputs to know if they belong to the same class or not. This loss function is also named Binary Cross-Entropy loss and owes its name to its nature based on setting up a binary classification problem between 2 classes for every class given. This loss is defined in the Figure 4.4.
- The ResNet model uses this loss function in its multi-class option, in which the input is a vector with sigmoid function values corresponding to each class; with which the loss is computed with the sum of the values obtained with the formula 4.1.

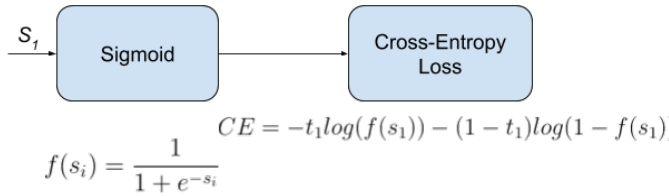

Figure 4.4: Softmax Cross-Entropy Pipeline [3].

Figure 4.4 represents the pipeline for each one of the C classes. It sets C independent binary classification problems (C'= 2), sums up the loss over the different binary problems, uses the gradients to backpropagate and the losses to monitor the global loss. The loss can be expressed as:

$$CE = \begin{cases} -log(f(s_1)) & if \quad t_1 = 1 \\ -log(1 - f(s_1)) & if \quad t_1 = 0 \end{cases} \tag{4.1}$$

Where $t_1 = 1$ means that the class coincides with the class i compared.

4.4 KNN over Siamese and ResNet18 feature vectors

After implementing both methods, the idea of checking the complementarity of the feature vectors processed by each model to improve the KNN classification came up. Thus, the following model was created:

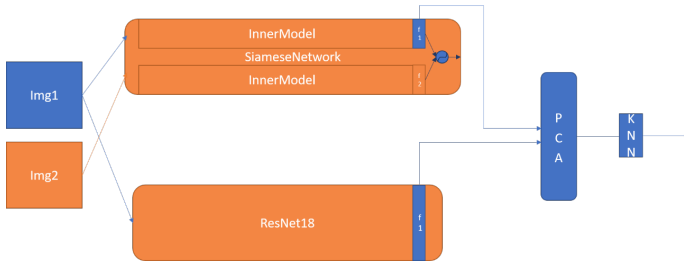

Figure 4.5: Siamese and Resnet feature Extraction Model with 1-NN Classifier .

The Figure 4.5 presents how these two previous models are combined in one and how the information flow looks. The parts filled in blue show from and to where the relevant information is processed. As can be observed, two images are needed as input due to the presence of the Siamese Network, but only the features extracted from the first picture (the one we want to classify) are going to be used in the 1-NN after processing both feature vectors with PCA to reduce the redundancies that can concur by mixing them.

CHAPTER 5

Experiments

5.1 Dataset

In order to train the models, two datasets have been used. MNist has performed the training on the Siamese, and ImageNet is used for training the ResNet18 as we are loading a pre-trained model.
To train the Siamese model, a modified version of the MNist dataset obtained from PyTorch database has been created. The iterator of this dataset returns a first image that will be the picture we want to classify and a second one that will work as a reference class, so the model can tell if the first belongs to the same category.
This data generator, when used for training purposes, is able to discriminate by classes. Therefore, the training dataset will lose one class. Being able, in this way, to not train the model with pictures from the class that we do not want the model to use.

5.2 Training

As we have different models, each one has its training method:

- The Siamese Network, as described in the dataset section, uses a data generator that trains the model selecting n-1 classes from the original dataset; forcing the model not to view any example of that class n.
  To train the Siamese Network, two types of Loss functions are implemented, Cross-Entropy Loss function and Contrastive Loss. Cross-Entropy loss, as can be seen in the figure 5.1, has a much faster convergence than the Contrastive Loss function (Figure 5.2) which actually does not converge. This points to a malfunction of the function probably due to the way it has been implemented.

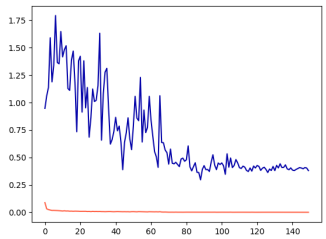

Figure 5.1: Cross-Entropy Loss,Blue:Validation. Red:Training.

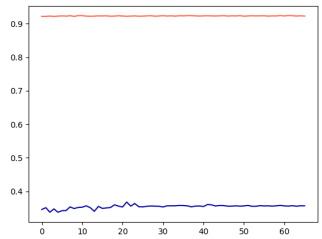

Figure 5.2: Contrastive Loss, Blue:Validation. Red:Training.

Therefore, the Cross-Entropy Loss function was the one selected during the training on the Siamese network.

The dataset used at Siamese training contains every class except the last one, 54060 pictures of numbers from zero to eight, and is validated using the excluded class (nine) in training. Furthermore, as the output from this Siamese model is not discreet, a threshold has been implemented.

The ResNet18 model used is a class already created in torch framework, being able to load a pre-trained version over ImageNet. This model expects a three channels image as input (RGB format), but the MNist dataset is codified as one channel images, so this input layer has been overwritten by a Conv-Layer of the same characteristics but which expects just one input channel. Additionally, the last output layer is replaced by a fully connected layer with 512 inputs and 10 outputs that will work as a classifier for the dataset.

The training of the ResNet model consists of two stages:

- The first one is done over the ResNet network. On account of the replacements of the input and output layers, the need to train these two new layers to fit them to the problem we want to solve (MNist Classification) appears. Therefore, a training stage is done over the ResNet network with every class of the dataset but the last one, as we want to preserve the One-shot learning environment.
- The second one works over KNN, where has been modified the ResNet layer. In this one, the different hyperparameters of the KNN have been modified for finding the best settings for the KNN model.

CHAPTER 6

Results

The results of the classification can be seen in the following table:

Model	**Hyperparameter**	**Accuracy**
Resnet18 on KNN	K = 1 Neighbour	0.5646
Siamese Network	Threshold = -5.7	0.9974
Siamese & Resnet18 Features on KNN	K = 1 Neighbour	0.5646

The above results refers the best results obtained from varying hyperparameters from the settings.

The KNN is set with 1 Neighbour because of the environment condition. Having one example of each class force the KNN classifier to compare at the training with a maximum of 1 Neighbour. If the K had been set as 2, probably half of the classes would not be represented by it.

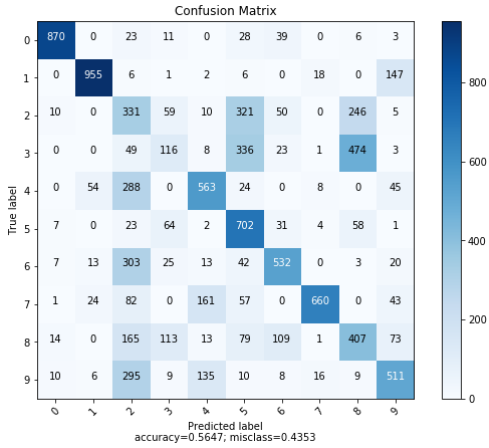

Figure 6.1: KNN Confusion Matrix.

Observing this confusion matrix, the classes that are more similar between each other show more cases of misclassification, causing a rate of *0.4353*. This is not an acceptable ratio as the model will be accurate approximately half of times, and a deep learning model usually expects to reach more than the 90 per cent of accuracy.

Attending the Siamese network, first of all, an analysis has been done to obtain the best threshold value that would perform on a maximum accuracy of the model; obtaining a value of *-5.7*.

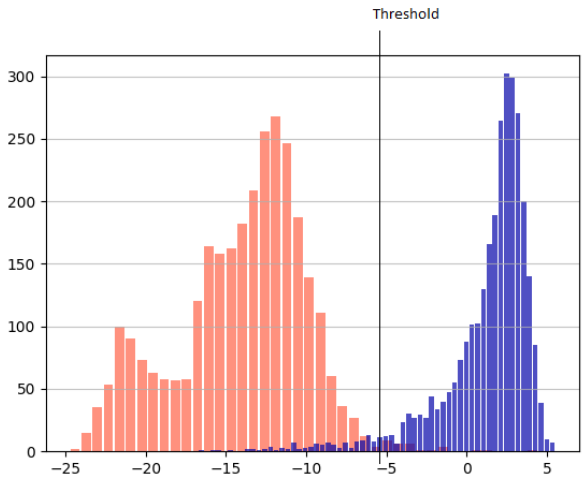

Figure 6.2: Histogram of similarity values and threshold applied of -5.7 to apply classification.

Once the threshold value was found, we proceed to test the model obtaining the confusion matrix we can observe down below. It has two squares representing if the example belongs to the 9^{th} class or owns to a different class.

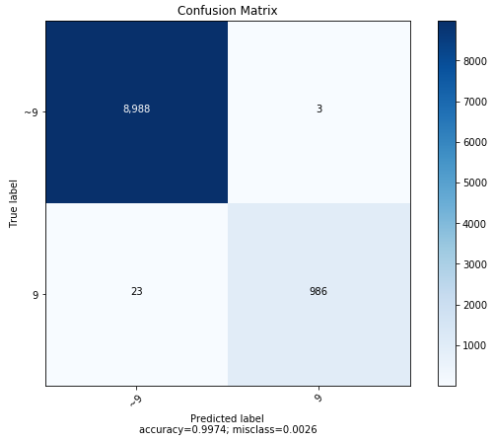

Figure 6.3: Confusion matrix of classification on class 9 or other.

As can be seen, the accuracy obtained in this model beats completely to the ResNet18 over a KNN model with the score of 0.9974. With this graph, we realise that just 3 cases out of 8.991 are misclassified into the new class added; giving high confidence rates to classes learned previously even if one new category has been added. Moreover, in just 23 out of 999 cases of this new class are being misled with other classes.

With the success of the Siamese network, the expectations over the mixed model were to obtain a bit of improvement when mixing both feature vectors, but the results given are the same than the previous model. Showing the conclusion of, that, at least, with the dataset we are working on, any new feature that could complement the KNN model is not found by the Siamese network.

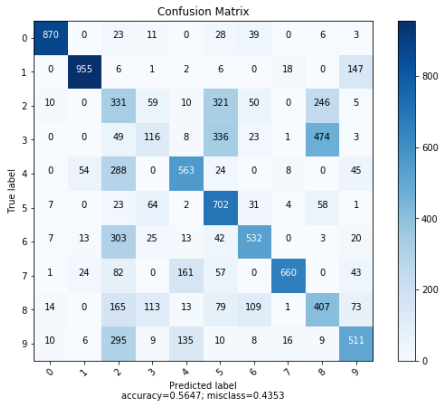

Figure 6.4: Confusion matrix Siamese and ResNet with 1-NN Classification.

CHAPTER 7

Conclusion

7.1 Discussion

This thesis has explored the concept of one-shot object recognition, giving more significant weight to the classifier task. For better understanding of the different approaches over one-shot learning, two different ways of classifying over this specific topic were implemented.

Looking at the results, it is evident that classification using Siamese networks will perform significantly better results than using Resnet18 embedding vectors over KNN. However, the difference between models relies on the computational cost and the usability of Siamese networks. Siamese network owes its similar cost to the embedding with KNN classifying system to discriminating just one unique class. To reach classifying purposes, the cost would increase theoretically by N times the current time (being N the number of classes we want to recognise). Since a single example would have to go through the Siamese network with N different pairs.

As it is shown in the images [][] showed in the Results part, the idea of mixing the embeddings extracted from each model, and processing them through PCA does not mean any improvement in the KNN classification.
Nonetheless, the idea of mixing embeddings extracted from both models, going through a PCA, in order to later classify them with a KNN does not improve the accuracy obtained from Resnet18 model. Thus, it can be deduced that both models are capable of extracting the same features from a simple dataset.

A good alternative to improve this project would be to take advantage of the accuracy of Siamese network to create a multi-classifier. The ability to discriminate examples from one class to another can be used to create a memory-based model in which, one example from each class would be processed during training and the feature vector extracted will be saved inside the model. In this way, when the model is used for predictions instead of returning a single value of similarity, it will return a vector with the similarity values for each class, being the class predicted the one that has a higher value than the threshold calculated in the training part. Applying probability, if we have 10 classes, the chances of classifying an example correctly would be theoretically:

$0.9974^{10} = 0.9743$

Which predict an excellent accuracy score.

7.2 Conclusion

In this thesis, two one-shot methods for object classification have been successfully implemented. Siamese models seems to be a suitable candidate for classifying objects that are not being seen during the training model giving an accuracy of almost the same as in training in regular conditions.

Both methods implemented in this project aim different goals. Despite the belief that the features extracted by both models were complementary, this fact has been discarded by the PCA processing and the KNN classification. This is due to the fact that the variance coefficient remains unchanged when features from the Siamese model are used together with the results from the ResNet model. However, it must be taken into account that this can be due to the simplicity of the image dataset used.

Bibliography

[1] Dominiqe Gilleman. *Convolutional Network (CNN)*. URL: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. (accessed: 16.07.2019).

[2] Dominiqe Gilleman. *Residual Net*. URL: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/residual_net.html. (accessed: 16.07.2019).

[3] Raul Gomez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/. (accessed: 18.07.2019).

[4] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Restarts". In: *CoRR* abs/1608.03983 (2016). URL: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

[5] Timothy Lillicrap" "Oriol Vinyals Charles Blundell. "Matching Networks for One Shot Learning". In: *cs.LG* abs/1608.03983 (2017). arXiv: 1608.03983. URL: https://arxiv.org/pdf/1606.04080.pdf.

[6] Sebastian Raschka. *ReLu Function*. URL: https://sebastianraschka.com/faq/docs/relu-derivative.html. (accessed: 05.07.2019).

[7] Suryansh. *Gradient Descent. All you need to know*. URL: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. (accessed: 16.07.2019).

[8] Suryansh. *Gradient Descent. All you need to know*. URL: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. (accessed: 05.07.2019).

[9] Wikipedia. *Hyperbolic Tangent*. URL: https://es.wikipedia.org/wiki/Tangente_hiperb%5C%C3%5C%B3lica. (accessed: 05.07.2019).

[10] Wikipedia. *Sigmoid Function*. URL: https://en.wikipedia.org/wiki/Sigmoid_function. (accessed: 05.07.2019).

{2.7182818284}ϑφερτυθιοπσδφγηξκλ